Piccole reti neurali crescono

Carlo Lucibello *Bocconi Institute for Data Science and Analytics, Università Bocconi, Milano, Italy*

Diamo qui un breve un accenno della fenomenologia delle *deep neural networks*. Vediamo poi come delle piccole reti, trattabili analiticamente attraverso le tecniche della fisica statistica, siano in grado di catturare un parte di questa fenomenologia.

Il *Loss landscape*

Le reti neurali artificiali sono indubbiamente la forza motrice che sta dietro al rapido sviluppo a cui abbiamo assistito negli ultimi 10 anni nel campo dell'intelligenza artificiale.

Numerosi sono i traguardi recentemente conseguiti che, oltre a motivare il sempre crescente interesse accademico, hanno affascinato, stupito e a volte forse anche intimorito il grande pubblico.

`Alpha-Go`, un algoritmo basato su *deep reinforcement learning* e su `Monte-Carlo` *tree search*, è riuscito nell'impresa che si riteneva ancora molto lontana di battere il campione mondiale di `Go`, Lee Sedol. Sedol, in seguito alla sconfitta, ha abbandonato il gioco competitivo. `Alpha-Go` ha poi lasciato spazio ad `Alpha-Zero`, una rete in grado di allenarsi da zero, ovvero senza supervisione umana ma solo competendo con se stessa.

`GPT-3`, un modello di generazione linguistica con 175 miliardi di parametri, è in grado di generare testi difficilmente distinguibili da quelli creati da uno scrittore umano.

Modelli generativi come le *Generative Adversarial Networks* (`GANs`) sono in grado di produrre immagini e video estremamente realistici. Questi modelli sono comunemente impiegati nella creazione di quelli che vengono chiamati i `Deep Fake`.

Nonostante le architetture neurali si facciano via via più complesse e diversi moduli vengano combinati in maniera sorprendente grazie alla fantasia dei ricercatori, la matematica ed i principi di funzionamento alla base rimangono molto semplici.

In via piuttosto generale, una rete neurale può essere pensata come una relazione tra *input* e *output*, ovvero una funzione, formata da una serie di operazioni algebriche seguite da operazioni *element-wise* non-lineari. Più precisamente, un rete neurale è caratterizzata da un insieme di parametri che chiameremo θ nel seguito, per cui è in grado di esprimere un'intera famiglia di funzioni $\{f_\theta\}$ al variare di tali parametri. L'**apprendimento** consiste proprio nel selezionare all'interno di questo vasto spazio dei parametri una configurazione che ben si adatti ai dati osservati.

Una caratteristica che ha contribuito al successo delle reti neurali, ed in particolare di quelle *deep*, e che le rende uno strumento molto versatile, è la loro espressività, ovvero la capacità di ben approssimare una arbitraria relazione di *input-output*.

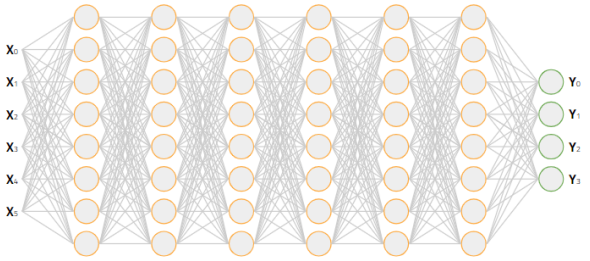


Figura 1: *Rappresentazione grafica di un multi-layer pereptron*

Il multilayer perceptron. Un esempio classico di *deep neural network* è il `multi-layer perceptron (MLP)`, rappresentato in Fig. 1. Un MLP con L *layers* è costituito da una matrice W_ℓ (i pesi sinaptici) e da un vettore $\boldsymbol{b}_\ell$ (il bias) per ogni *layer* ℓ. Inoltre ad ogni *layer* è associata una funzione σ_ℓ che opera *element-wise* e che viene detta funzione di attivazione. Per ogni dato vettore di *input* $\boldsymbol{x}$, la rete fornirà in *output* una predizione $\hat{y}$ ottenuta attraverso il cosiddetto **forward pass**:

$$\begin{aligned} \boldsymbol{x}_1 &= \sigma_1(W_1\,\boldsymbol{x} + \boldsymbol{b}_1) \\ \boldsymbol{x}_2 &= \sigma_2(W_2\,\boldsymbol{x}_1 + \boldsymbol{b}_2) \\ &\vdots \\ \hat{y} &= \sigma_L(W_L\,\boldsymbol{x}_{L-1} + \boldsymbol{b}_L) \end{aligned}$$

Chiamando θ l'insieme dei pesi e dei *bias*, abbiamo così definito la nostra funzione di predizione $\hat{y} = f_\theta(\boldsymbol{x})$. In uno scenario di apprendimento supervisionato, si ha a disposizione un *dataset* D di esempi costituiti da coppie di *input* $\boldsymbol{x}$ ed etichette y. Il problema dell'apprendimento è formulato come problema di ottimizzazione di una **Loss function** $\mathcal{L}(\theta)$ che assume la forma

$$\mathcal{L}(\theta) = \mathbb{E}_{(\boldsymbol{x},y)\sim D}\ \ell\left(y, f_\theta(\boldsymbol{x})\right) \tag{1}$$

Qui la funzione $\ell(y,\hat{y})$ è tipicamente l'errore quadratico medio nei *task* di regressione e la *cross-entropy* nei *task* di classificazione.

Ottimizzare la Loss. L'ottimizzazione viene conseguita attraverso varianti del semplice algoritmo iterativo noto come **gradient descent** (GD):

$$\theta^{t+1} = \theta^t - \eta\nabla\mathcal{L}(\theta^t) \tag{2}$$

Difatti, nonostante una pletora di metodi più sofisticati, quali ad esempio la famiglia dei metodi di `quasi-Newton`, siano stati prodotti dalla comunità di ottimizzazione, l'altissima dimensionalità del problema rende computazionalmente proibitivo ogni approccio significativamente più complesso del semplice GD. Inoltre, per guadagnare in efficienza computazionale e far fronte all'enorme aumento della dimensione dei *dataset* usati per il *training*, una variante dello schema di GD, nota come **Stochastic Gradient Descent** (SGD) viene di fatto utilizzata. Nello schema di SGD, un sottoinsieme casuale degli esempi B_t, detto **mini-batch**, viene estratto ad ogni iterazione ed usato per calcolare una versione approssimata della `Loss` (1) che a sua volta fornirà una approssimazione del gradiente usato nell'update di Eq. (2). La dinamica di SGD è quindi definita da:

$$\theta^{t+1} = \theta^t - \eta\nabla\mathcal{L}_{B_t}(\theta^t) \tag{3}$$

Alla fine del `training`, la rete viene testata su un *set* di esempi escluso dal *training set*, in modo da valutare l'effettiva capacità della rete di generalizzare, ovvero rispondere correttamente in situazioni nuove ma simili a quelle di cui si è fatta esperienza. Questo è il criterio che anche in termini comuni useremmo per giudicare un apprendimento di successo. Ed è proprio qui, nella straordinaria capacità di generalizzare, nonostante il grande numero di parametri che caratterizzano questi modelli, che sta un'altra delle chiavi del successo del `deep learning`.

Ora che abbiamo definito il nostro *setting*, è interessante fare una serie di considerazioni.

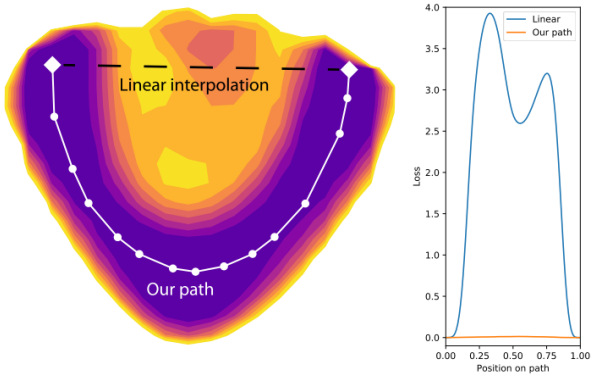
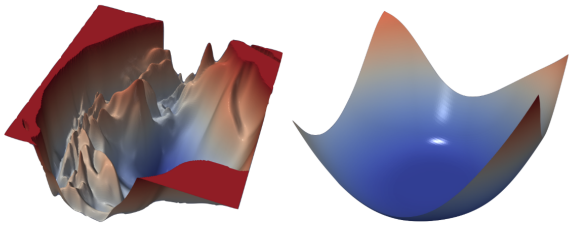


Figura 2: *Proiezione in due dimensioni di un percorso a bassa training loss che connette due soluzioni algortimiche. Immagine da Ref. [1]*

Minimi locali, minimi globali. La `Loss` di Eq. (1) è generalmente una funzione altamente non-convessa. Il suo *landscape* potrebbe quindi essere molto complesso e possedere un numero molto grande di minimi locali e di selle, anche a valori di `loss` alti. L'evidenza empirica però dimostra che reti sufficientemente grandi e allenate con SGD arrivano facilmente a valori di loss molto bassi. Non rimangono quindi incastrati in minimi locali. Mentre questo fenomeno era stato inizialmente attribuito allo stocasticità intrinseca di SGD, potenzialmente in grado di permettere al sistema di evadere dai minimi locali, l'evidenza corrente è che questa sia una proprietà propria del *landscape* e delle condizioni di inizializzazione. Se minimi locali alti esistono, non sembrano essere algoritmicamente rilevanti. Inoltre, per alcune architetture e sotto assunzioni molto forti e spesso irrealistiche, si hanno dei risultati che escludono la presenza di minimi locali che non siano anche globali.

Un fondo connesso e rugoso. L'evidenza sperimentale suggerisce che i minimi algoritmicamente accessibili possono essere connessi lungo traiettorie a bassa `loss` (si veda Fig. 2, sinistra). Generalmente questi percorsi sono tortuosi: una semplice interpolazione lineare tra due minimi darà l'impressione di trovarsi di fronte a delle barriere (Fig. 2, destra).



(a) ResNet-110, no skip connections (b) DenseNet, 121 layers

Figura 3: *L'uso di connessioni residue contribuisce a smussare e appiattire il landscape della training loss. Immagine da Ref. [2].*

Minimi larghi, minimi stretti, generalizzazione. Scelte architetturali, quali ad esempio il numero di layer, la loro larghezza, l'impiego di connessioni residue, hanno un enorme impatto sulla geometria della `loss`, come ad esempio mostrato in Fig. 3. Alcuni iperparametri che regolano il *training*, in particolare la *batch size*, il *learning rate* e l'utilizzo del *dropout*, influenzano la piattezza della `loss` intorno alle soluzioni trovate. Ciò ha un impatto rilevante sulle proprietà di generalizzazione. L'evidenza numerica mostra che minimi larghi hanno migliori proprietà di generalizzazione di quelli stretti.

La dinamica. L'analisi della dinamica di GD, e ancor di più di SGD, è sostanzialmente intrattabile, complicata dall'alta dimensionalità e dalla generale non-convessità del problema. SGD è in prima approssimazione simile ad una dinamica di Langevin, seppur vi entri un rumore non-bianco e dipendente dalla posizione. Misure sperimentali delle funzioni di correlazione a due punti nella dinamica del *training* ci mostrano delle connessioni con le dinamiche lente proprie dei sistemi fisici vetrosi [3].

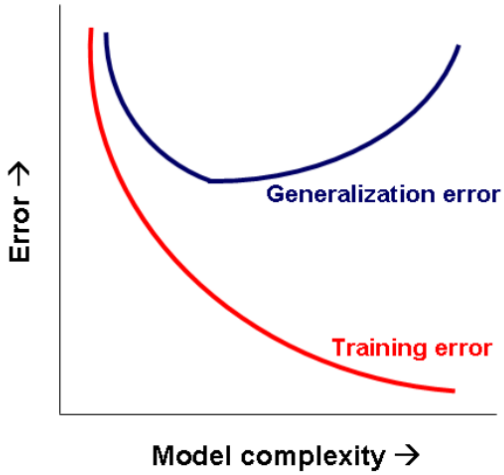


Figura 4: *Il problema dell'overfitting nella teoria statistica classica*

La teoria classica dell'apprendimento statistico. La teoria classica dell'apprendimento statistico è in forte difficoltà nello spiegare il successo delle reti neurali. Infatti, tale teoria prevede che quando il numero di parametri N del modello diventa molto maggiore del numero dei dati di training P, $N \gg P$, si va incontro al fenomeno dell'*overfitting* (si veda Fig. 4). Si dovrebbe quindi osservare un aumento delle errore di generalizzazione all'aumentare di N. In pratica invece questo non si asserva per le `deep networks`. Queste tipicamente operano nel regime $N \gg P$,

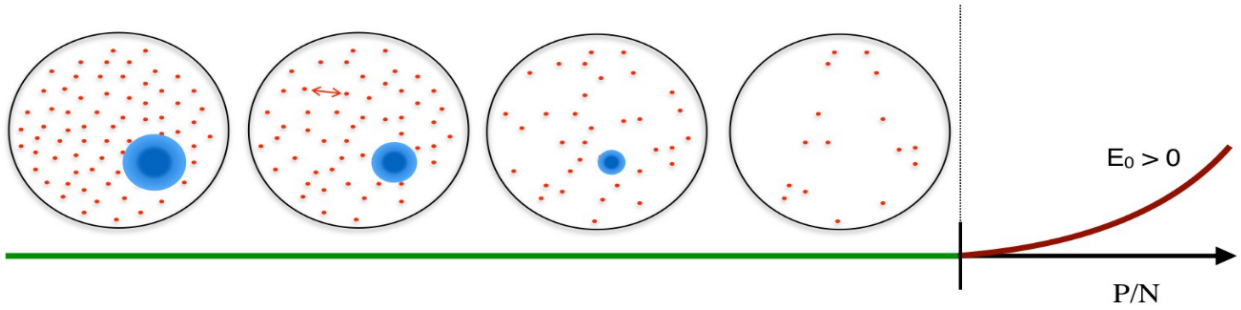


Figura 5: *Lo spazio delle soluzioni nel percettrone binario. La gran parte delle soluzioni sono isolate, ma un numero minore (ma pur sempre esponenziale) forma una regione densa e connessa rappresentata in blu.*

dove i *bound* classici sull'errore di generalizzazione diventano vacui. Tra le sfide concettuali più rilevanti in questo campo vi è il capire la natura di questo fenomeno, la sua relazione con la scelta delle architetture, l'inizializzazione delle reti e la dinamica del *training*. Di enorme rilevanza è l'individuare dei *bound* per l'errore di generalizzazione al tempo stesso più stringenti e computazionalmente trattabili . Il sacro Graal qui consiste nel rendere esplicita una qualche regolarizzazione implicita che agisce in questi modelli e che ne riduce il numero di parametri efficaci, comprimendo lo spazio delle ipotesi.

Il percettrone binario

Dopo questo *excursus* nel mondo delle reti neurali profonde, ci spostiamo ora su uno scenario molto più ristretto, quello della rete neurale minimale, ovvero il singolo neurone, chiamato in questo contesto **percettrone**. Il *task* è quello di classificazione binaria, in due classi che denotiamo con $y = +1$ e $y = -1$. La funzione di predizione associata al percettrone è quindi

$$\hat{y} = \text{sign} \sum_{i=1}^{N} W_i \, x_i^\mu \tag{4}$$

Se i pesi W_i sono lasciati liberi di assumere valori arbitrari nel continuo, si può dimostrare che il problema, in presenza di un *training set* linearmente separabile, ammette un insieme di soluzioni convesso, quindi poco interessante e non rappresentativo di quanto avviene in reti più grandi. Consideriamo invece il caso in cui i pesi possono assumere solo valori binari, $W_i \in \{-1, +1\}$. Andremo a vedere che questo sistema è dotato di una fenomenologia molto ricca. Computazionalmente, il problema diventa più duro (seppur ancora algoritmicamente risolvibile [6]), in quanto non ci si potrà avvalere degli strumenti di ottimizzazione basati sul gradiente. Questo modello, chiamato **percettrone binario**, è stato analizzato per lungo tempo dalla comunità di fisica statistica [4, 5].

Più in generale, la possibilità di allenare reti neurali binarie è argomento di vivace ricerca nella comunità del *machine learning*, in quanto ha notevoli ricadute in termini di efficienza computazionale ed energetica, nonché ai fini della compressione e dell'utilizzo di rete neurali su piccoli dispositivi con scarsa risorse di memoria e computazionali.

Proseguiamo la nostra analisi considerando un *setting* molto semplice, in cui P esempi di *training* sono forniti alle reti. Un esempio è formato da una coppia x^μ e y^μ generata in modo casuale, con un *output* scorrelato dall'*input*. Non c'è quindi in questo caso una regola da imparare (ne forniremo invece una nella prossima sezione) ma siamo solamente interessati ad analizzare il numero di soluzioni di questo *constraint satisfaction problem*. Tale numero è dato dalla funzione di partizione Z associata al problema, definita da

$$Z = \sum_{\boldsymbol{W}} \prod_{\mu=1}^{P} \mathbb{I}\left(y^\mu \sum_{i=1}^{N} W_i \, x_i^\mu > 0 \right) \tag{5}$$

Usando il metodo delle repliche ed il metodo della cavità della teoria dei vetri di spin, è possibile calcolare l'entropia media del sistema nel limite

termodinamico:

$$S = \lim_{N\to\infty} \frac{1}{N} \mathbb{E} \log Z \tag{6}$$

Qui l'aspettazione è sulle realizzazione del *training set* ed il limite viene preso andando a guardare al regime non triviale in cui il numero di esempi rimane proporzionale ad N, ovvero fissiamo una costante α per cui $P = \alpha N$. Il calcolo dell'entropia S, unita ad un'analisi delle *performance* degli algoritmi, rivela una serie di nette transizioni di fase al variare di α in corrispondenza di alcuni valori che chiamiamo α_{alg} e α_c.

- Per $\alpha < \alpha_{alg}$ esiste quasi sempre un numero esponenziale di soluzioni ed è algoritmicamente semplice trovare una di queste. Come vedremo in seguito però, non tutte le soluzioni sono algoritmicamente accessibili (infatti gran parte non lo sono). Il valore di α_{alg} dipende dall'algoritmo considerato.

- Per α intermedio invece, $\alpha_{alg} < \alpha < \alpha_c$, nonostante la presenza di un numero esponenziale di soluzioni, nessun algoritmo noto riesce a raggiungere una soluzione in tempo polinomiale. In questo regime il problema è `duro`.

- Per $\alpha > \alpha_c$ il problema con alta probabilità non ammette soluzioni. Qualsiasi configurazione W classifica male un sottoinsieme degli esempi del *training set*.

Si veda Fig. 5 per una rappresentazione del *set* di soluzioni al variare di α.

Lo scenario Insegnante-Allievo

Arricchiamo ora il problema considerando un modello di generazione dei dati in cui gli *input* $\boldsymbol{x}^\mu$ continuano ad essere *random*, ma in cui le etichette y^μ sono generate da un secondo percettrone, preso con pesi *random*, che chiameremo `l'Insegnante`. Il percettrone su cui viene fatto l'apprendimento sarà chiamato lo `Studente`.

Anche questo contesto è analizzabile attraverso le tecniche della teoria degli *spin-glass*. Possiamo a questo punto porci due domande, strettamente connesse tra di loro: quanto è bravo lo `Studente` a generalizzare, ovvero a classificare correttamente nuovi esempi prodotti dall'`Insegnante`? Siamo capaci di inferire, visti $P = \alpha N$ esempi, il vettore di pesi dell'`Insegnante`?

La risposta dipende dal regime in α in cui operiamo. Ancora una volta abbiamo una sequenza di transizioni, a dei valori di α che chiamiamo α_{IT} e α_*.

- Per $\alpha < \alpha_{IT}$ esiste un numero esponenziale di soluzioni al problema, tra cui l'`Insegnante` stesso. In questo mare di soluzioni non è possibile discriminare l'`Insegnante`. In questo regime l'inferenza perfetta è impossibile dal punto di vista della teoria dell'informazione. Si nota in questo regime una discrepanza tra l'errore di generalizzazione predetto dalla teoria per una soluzione tipica (campionata uniformemente a caso) e quello fornito dalle soluzioni trovate dagli algoritmi, essendo quest'ultimo sistematicamente migliore. Scopriremo nella prossima sezione il perché.

- Per α tale che $\alpha_{IT} < \alpha < \alpha_*$, l'Insegnante si trova ad essere l'unica soluzione, quindi in linea di principio sarebbe possibile farne l'inferenza perfetta e arrivare ad errore di generalizzazione nullo. Tuttavia non sono noti algoritmi efficienti in grado di trovare l'Insegnate, il problema è algoritmicamente duro.

- Per $\alpha > \alpha_*$ l'`Insegnante` si trova ad essere l'unica soluzione e l'inferenza perfetta è ottenibile in tempo polinomiale. Il problema è semplice in questo regime.

Recentemente questo scenario è stato rigorosamente comprovato, confermando le predizioni delle teoria delle repliche [7].

La geometria dello spazio delle soluzioni

In entrambi i problemi analizzati, la descrizione *coarse grained* data dallo studio dell'entropia del sistema, seppur in grado di mettere alla luce delle sorprendenti transizioni di fase, non è in grado di spiegare appieno le *performance* degli algoritmi e la natura delle soluzioni trovate. Quello che

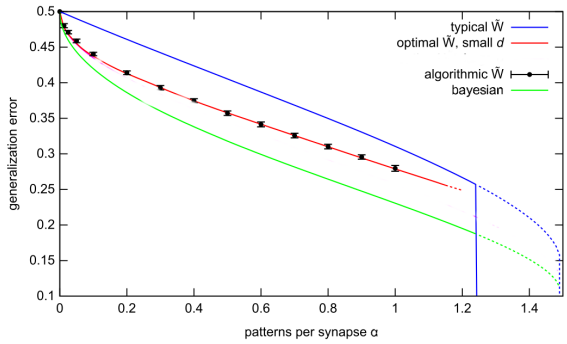


Figura 6: *Errore di generalizzazione teorico predetto per le soluzioni isolate del percettrone binario, comparato a quello delle soluzioni all'interno del cluster denso e a quello dato da soluzioni trovate da algoritmi polinomiali. Questi ultimi due coincidono perfettamente.*

succede infatti, è che le soluzioni trovate dagli algoritmi polinomiali quali `Belief Propagation` (BP), ovvero le soluzioni algoritmicamente accessibili, non corrispondono a soluzioni estratte uniformemente a caso dall'insieme delle soluzioni, ma piuttosto da una misura con un forte *bias* verso alcune regioni.

Un'idea più precisa possiamo farcela andando a vedere nel dettaglio la geometria locale dello spazio delle soluzioni, grazie ad una tecnica proposta da Parisi e Franz nel contesto degli *spin-glass* [8].

Chiamiamo entropia locale $S_{loc}(\tilde{\boldsymbol{W}}, d)$ di una configurazione $\tilde{\boldsymbol{W}}$, il (log-)numero di soluzione del problema entro una distanza intensiva d da $\tilde{\boldsymbol{W}}$, ovvero

$$S_{loc}(\tilde{\boldsymbol{W}}, d) = \frac{1}{N} \log \left[\sum_{\boldsymbol{W}} \prod_{\mu=1}^{P} \mathbb{I}\left(y^\mu \sum_{i=1}^{N} W_i \, x_i^\mu > 0 \right) \times \mathbb{I}\left(\|\boldsymbol{W} - \tilde{\boldsymbol{W}}\| < dN \right) \right] \tag{7}$$

Il calcolo dell'entropia locale per il percettrone, portato avanti in Ref. [9], svela una sorprendente geometria e ci permette di chiarificare le *performance* algoritmiche. Risulta infatti che la gran parte delle soluzioni del sistema è isolato, ovvero si trova ad essere a distanza estensiva da altre soluzioni. Affianco a queste soluzioni dominanti, un numero sempre esponenziale ma sottodominante è raggruppato in un cluster ad alta entropia locale, denso e connesso. Evidenze analitiche e numeriche mostrano che questo *cluster* smette di esistere in corrispondenza delle transizioni algoritmiche (si veda Fig. 5). In altre parole, nei regimi sotto α_c e sotto α_{IT} nei due scenari esaminati, gli algoritmi trovano soluzioni in tempo polinomiale fintanto che il *cluster* denso esiste, e le soluzioni che trovano vivono appunto all'interno di una regione densa di soluzioni. Le soluzioni isolate risultano algoritmicamente inaccessibili.

Abbiamo quindi un *landscape* simile a quello di un campo da golf, arricchito però dalla presenza di una grossa e profonda buca che rende possibile l'apprendimento.

Altro risultato dell'analisi è che le soluzioni all'interno del *cluster* denso generalizzano meglio delle soluzioni isolate (Fig. 6). Abbiamo quindi ritrovato in questo semplice modello l'analogo del *gap* di generalizzazione che si osserva tra minimi larghi e stretti nelle `deep network`.

Conclusioni

In questo articolo, abbiamo brevemente riassunto alcuni tratti fenomenologici che si osservano empiricamente nelle `deep network`, con particolare attenzione al *landscape* della `Loss function` e alle proprietà di generalizzazione dei diversi minimi algoritmicamente accessibili.

Abbiamo poi analizzato un modello di rete neurale analiticamente trattabile, il percettrone binario, ed evidenziato la stretta connessione tra *performance* algoritmiche e geometria dello spazio delle soluzioni. L'intuizione che ne deriva è estendibile al contesto del `deep learning`, ed ha portato all'elaborazione di algoritmi che per costruzione sono indirizzati verso minimi larghi e ad alta entropia locale, con buone proprietà di generalizzazione [10, 11].

Sebbene questo tipo di analisi su modelli semplici porti alle creazione di framework interpretativi molto generali ed anche a ricadute pratiche nel contesto deep, alcuni limiti sono evidenti. Le complesse architetture odierne sono largamente fuori scala per le tecniche di analisi teorica della fisica statistica. Inoltre le assunzioni di dati i.i.d. comunemente usate sono fortemente irrealistiche. Su quest'ultimo fronte, degli importanti passi avanti si sono visti di recente con l'introdu-

zione di modelli generativi di dati strutturati ma analiticamente trattabili [12].

Le piccole reti neurali sono cresciute a dismisura nel corso degli anni, c'è bisogno di nuovi strumenti di analisi teorica per continuare a fare strada assieme.

[1] F. Draxler et al.: *Essentially No Barriers in Neural Network Energy Landscape*, Proceedings of the 35th International Conference on Machine Learning, PMLR 80 (2018) 1308.

[2] H. Li et al.: *Visualizing the Loss Landscape of Neural Nets*, Advances in Neural Information Processing Systems 31 (NIPS 2018) (2018).

[3] M. Baity-Jesi et al. *Comparing dynamics: Deep neural networks versus glassy systems*, Proceedings of the 35th International Conference on Machine Learning, PMLR 80 (2018) 314.

[4] W. Krauth, M. Mézard: *Storage capacity of memory networks with binary couplings*, Journal the Physique, 50 (1989) 3057.

[5] A. Engel, C. Vand der Broeck: *Statistical mechanics of learning* Cambridge University Press, Cambridge (2001).

[6] A. Braunstein, R. Zecchina: *Learning by message passing in networks of discrete synapses*, Physical Review Letters, 96 (2016) 030201.

[7] J. Barbier et al.: *Optimal errors and phase transitions in high-dimensional generalized linear models*, PNAS, 116 (2019) 5451.

[8] S. Franz, G. Parisi: *Recipes for metastable states in Spin Glasses*, Journal de Physique, 5 (1995) 1401.

[9] C. Baldassi et al.: *Subdominant Dense Clusters Allow for Simple Learning and High Computational Performance in Neural Networks with Discrete Synapses*, Physical Review Letters, 115 (2015) 128101.

[10] P. Chaudhari et al. : *Entropy-SGD: Biasing Gradient Descent Into Wide Valleys*, ICLR 2017., arXiv:1611.01838.

[11] C. Baldassi et al.: *Unreasonable effectiveness of learning neural networks: From accessible states and robust ensembles to basic algorithmic schemes*, „ PNAS (113) 2016.E7655

[12] S. Mei, A. Montanari: *The generalization error of random features regression: Precise asymptotics and double descent curve*, arXiv:1908.05355

Carlo Lucibello: è Assistant Professor presso l'università Bocconi. Si occupa di problemi all'interfaccia tra il machine learning e la fisica statistica.